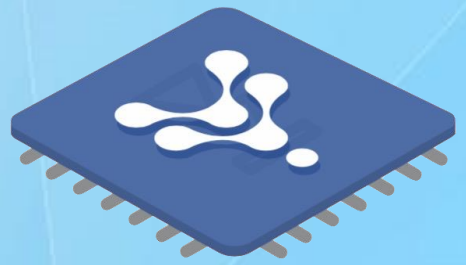
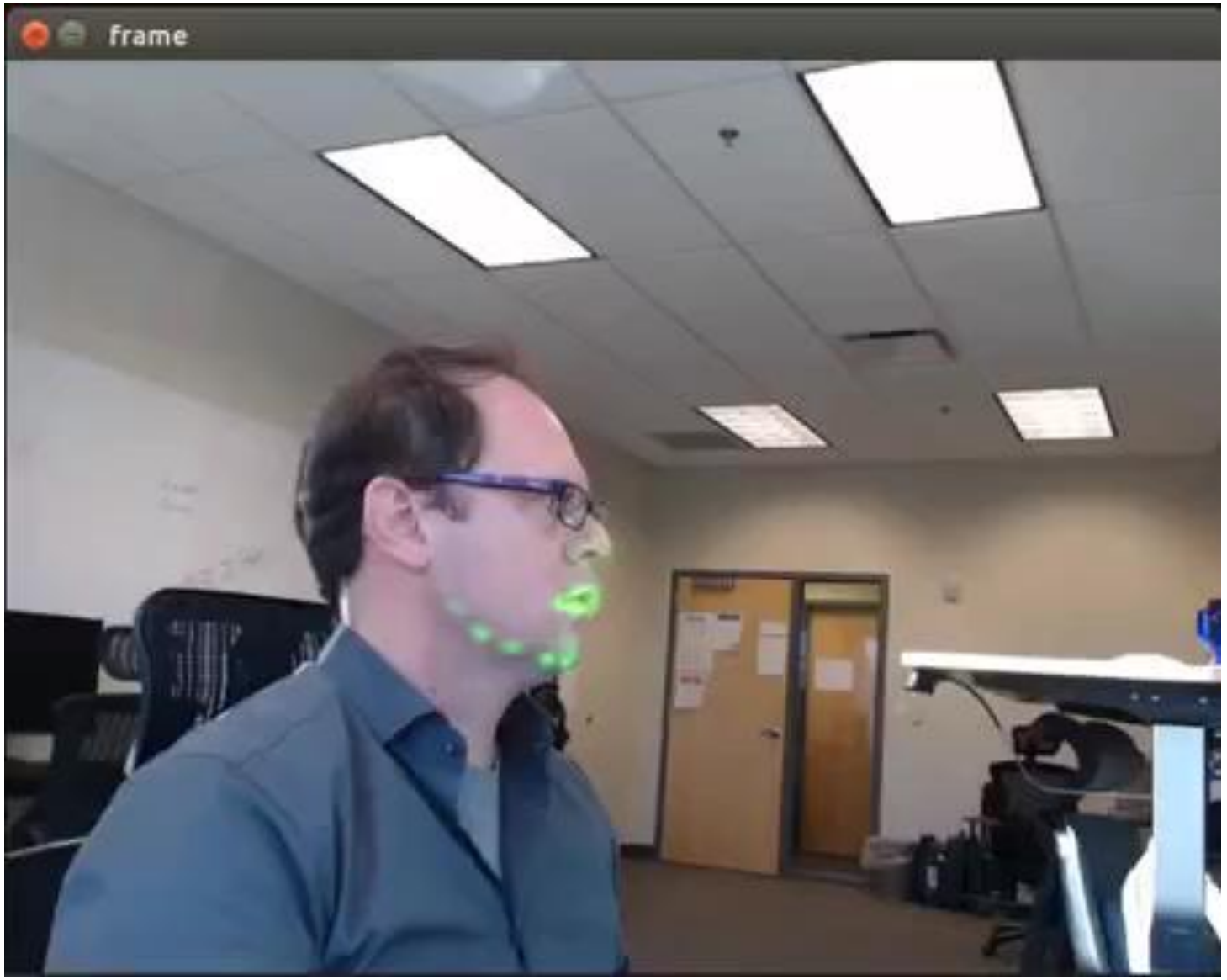
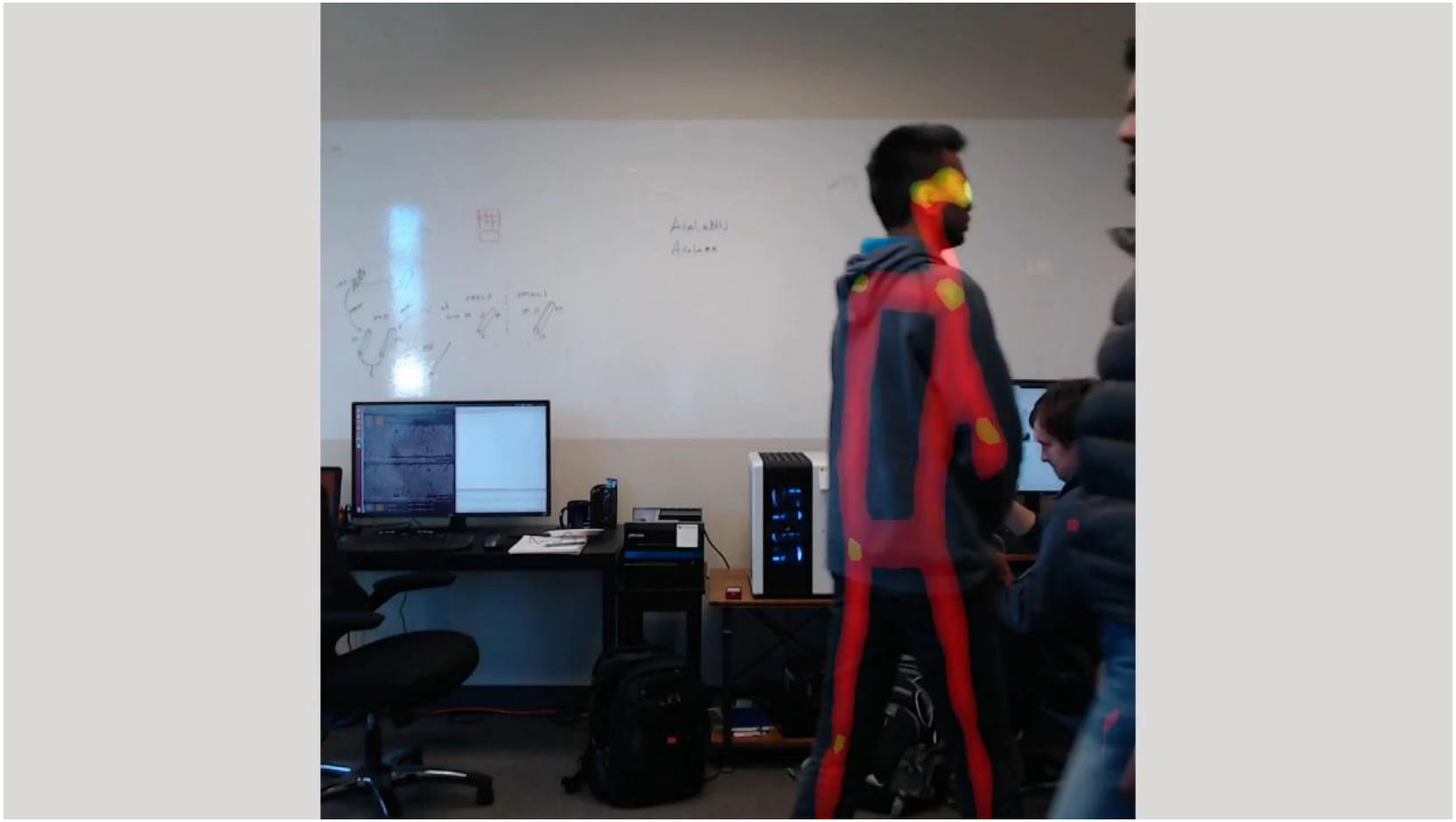


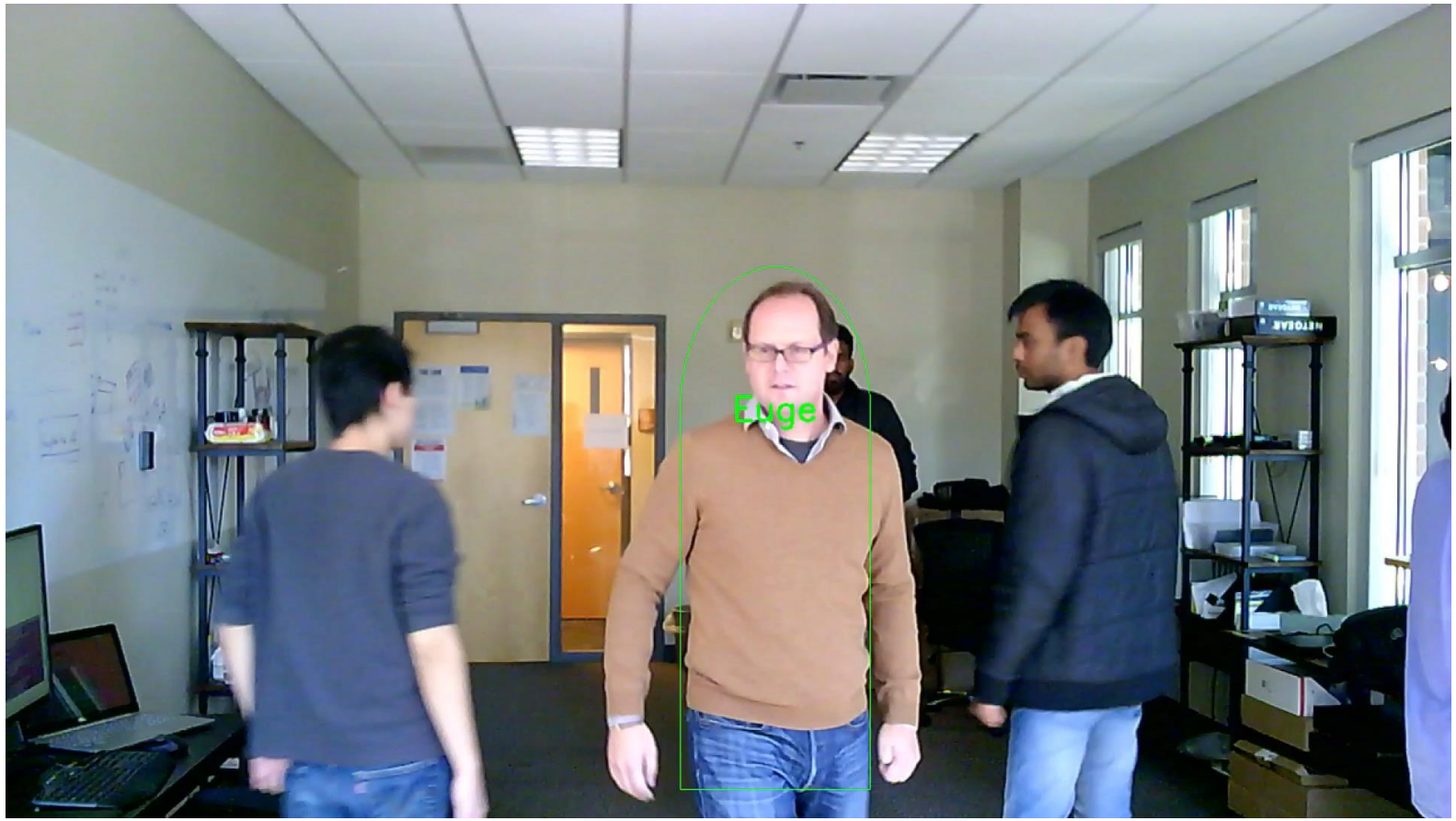
FW NXT

Efficient compiler code generation for Deep Learning Snowflake co-processor

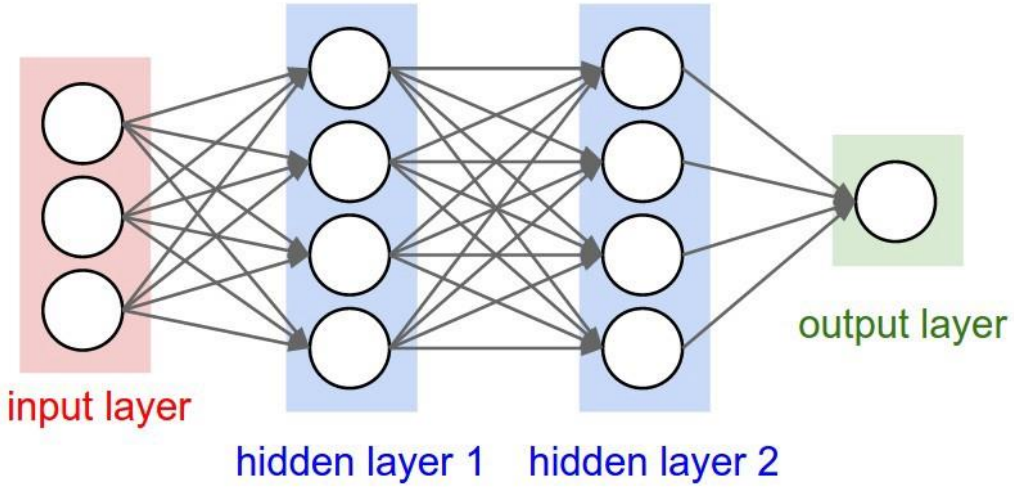
Deep Learning

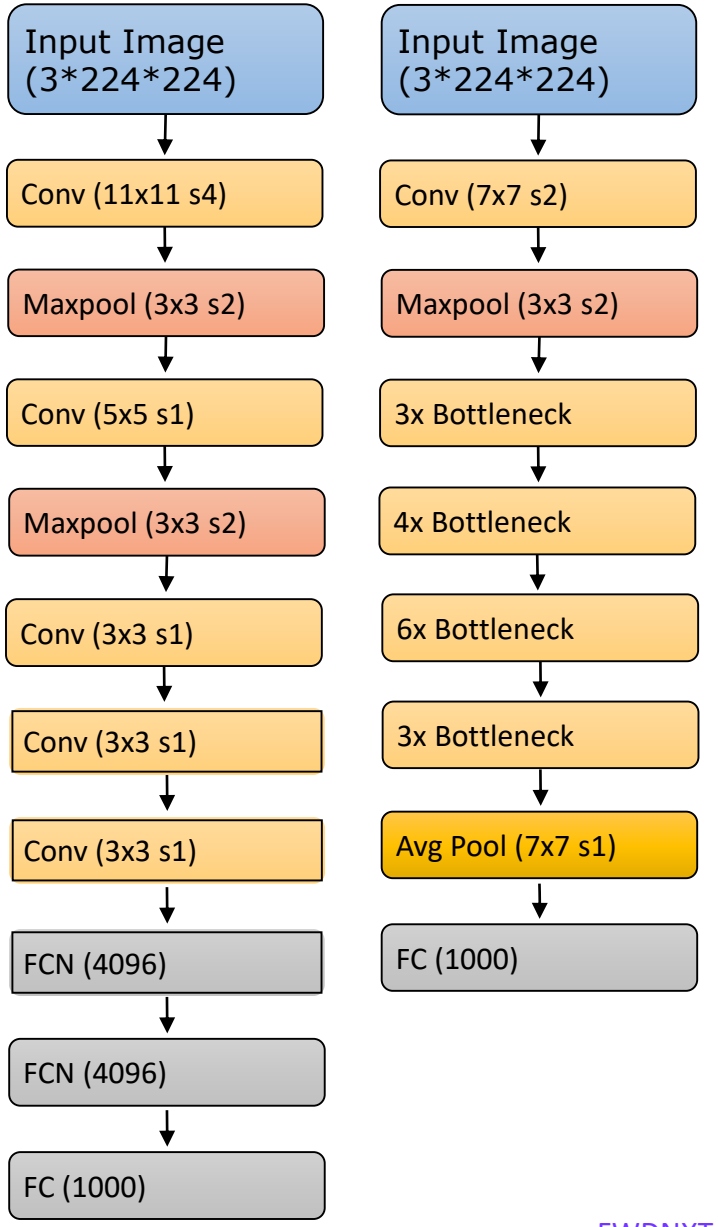


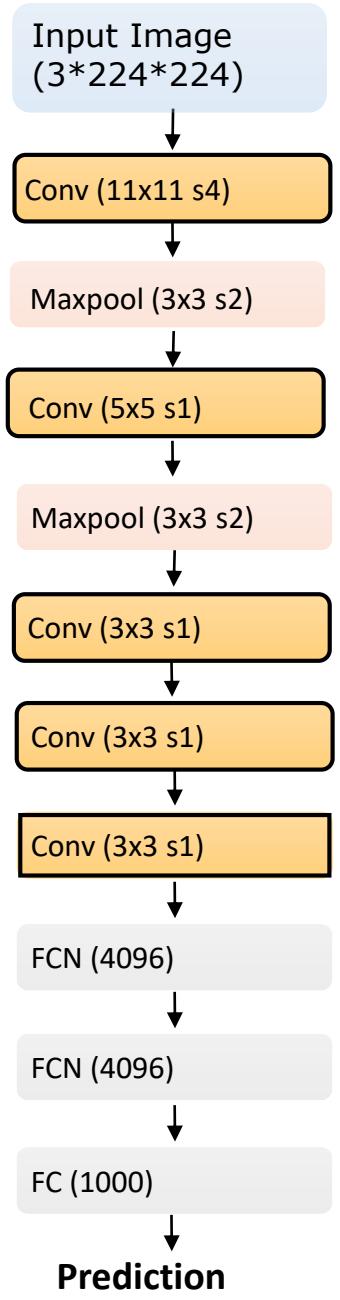


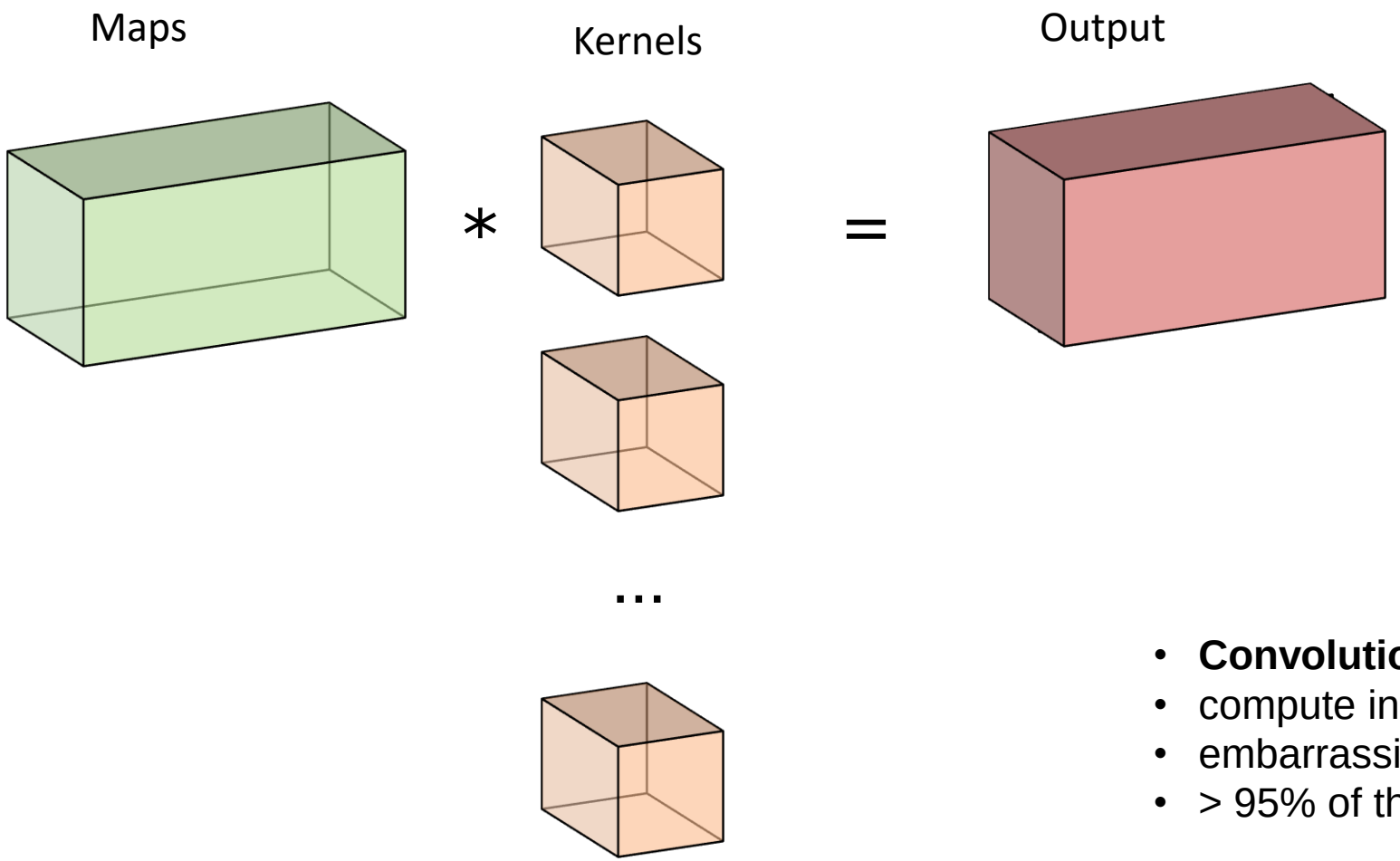


Conv Neural Networks



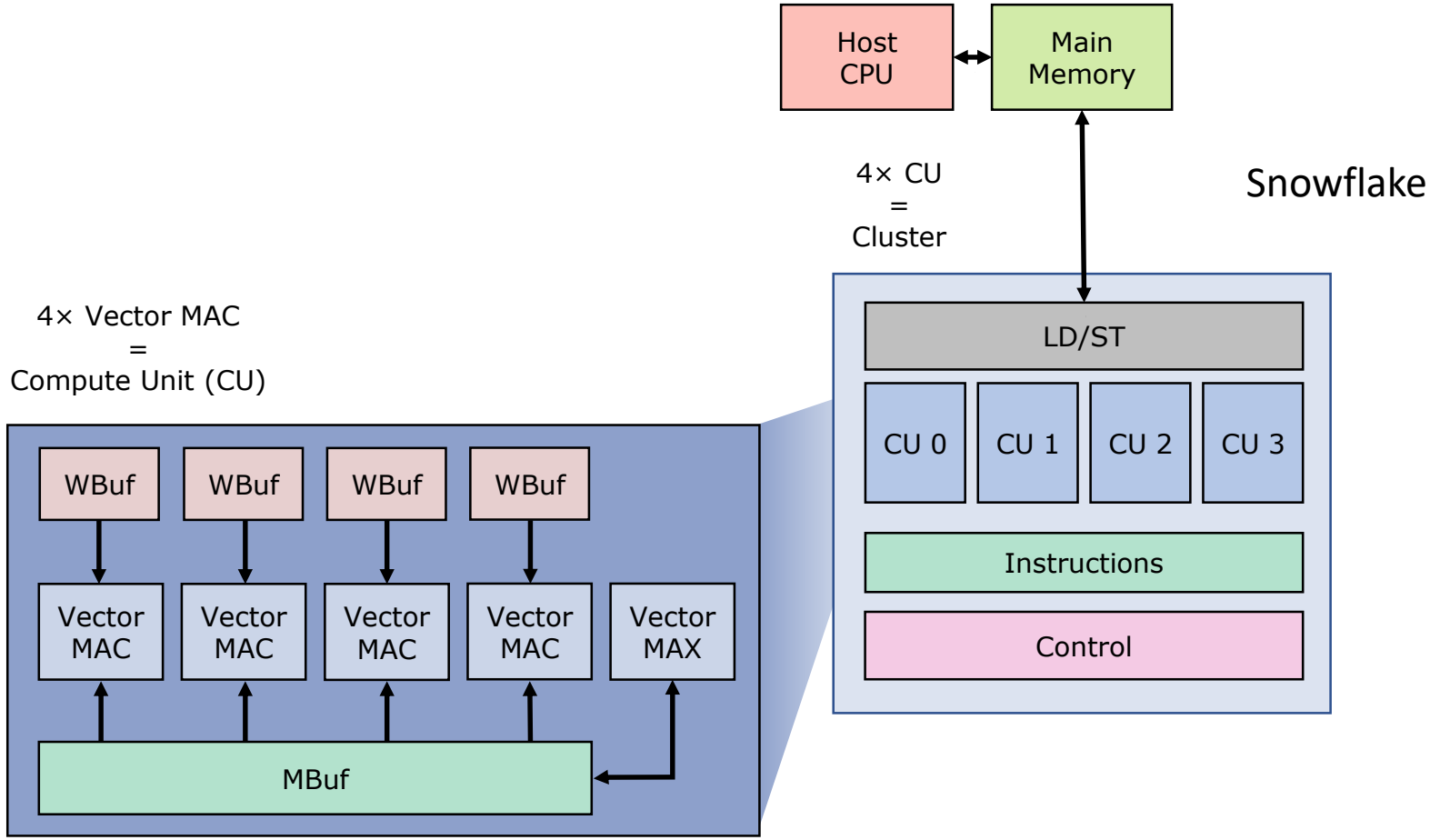






- **Convolutions**:
- compute intensive
- embarrassingly parallel
- > 95% of the workload

Co-processor



Snowflake instruction set

Move

Add

Multiply

Branch

MAC : Multiply and accumulate a trace

MAX: Compare a trace

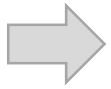
VMOV: Pre-load values

LOAD: mem2buffer

TMOV: buffer2mem

Goal

```python
import torch
import torch.nn as nn
class ModelDef(nn.Module):
    def __init__(self, num_classes=1000):
        super(ModelDef, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Conv2d(256, 4096, kernel_size=6),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Conv2d(4096, 4096, kernel_size=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(4096, num_classes, kernel_size=1),
            # nn.Softmax(),
        )
    def forward(self, x):
        x = self.features(x)
        x = self.classifier(x)
        return x
```

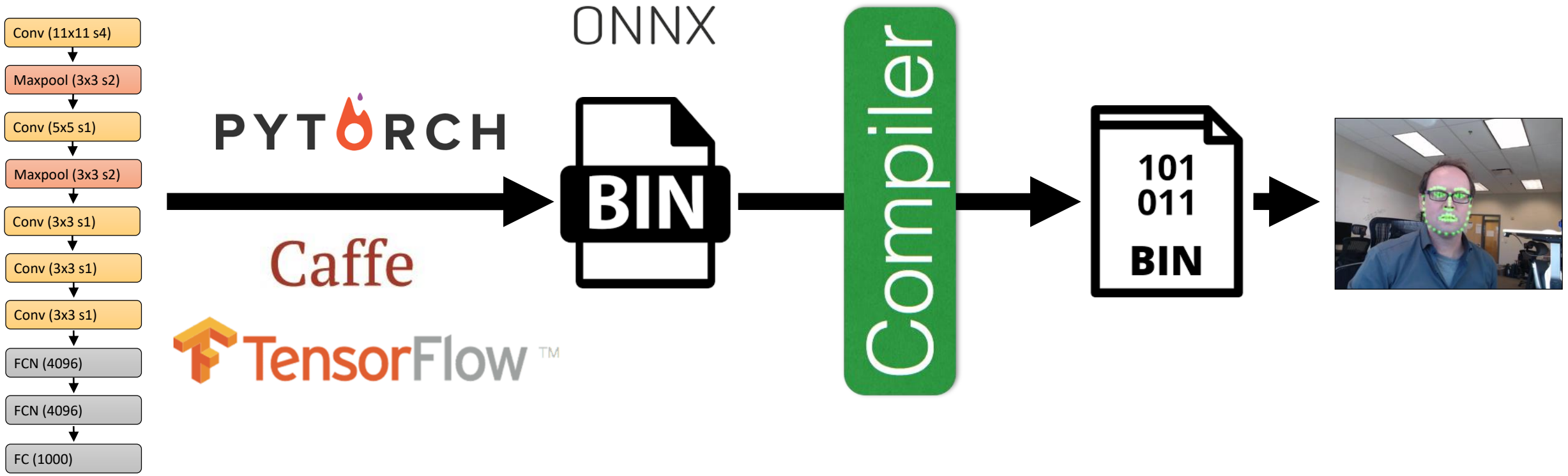


```
comp_add(0,1,22,0,0x00000f);
data_mov(1,18,1,0x000000);
comp_add(0,0,9,0,0x000000);
data_mov(0,3,0,0x3fffff);
data_mov(0,2,0,0x3fffff);
comp_mac(1,0x1f,0,1,0x000300);
brch_inst('l',21,2,0x000fff);
comp_add(0,0,0,0,0x001380);
comp_add(0,1,1,0,0x004800);

...
```

- Correctness
- Speed
- Extendibility

Overview



neural network

training frameworks

network description

Snowflake compiler

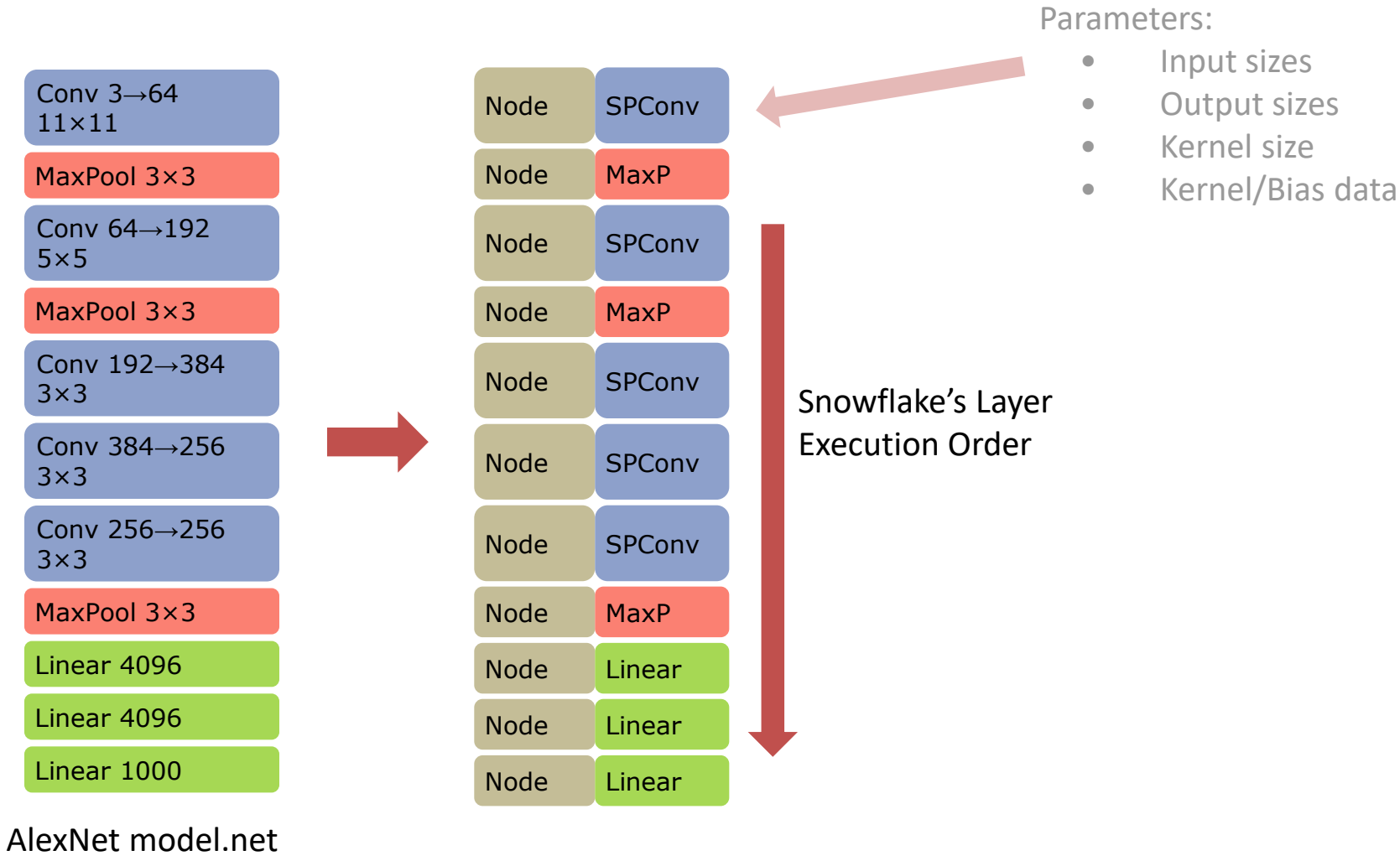
Snowflake code

deployment

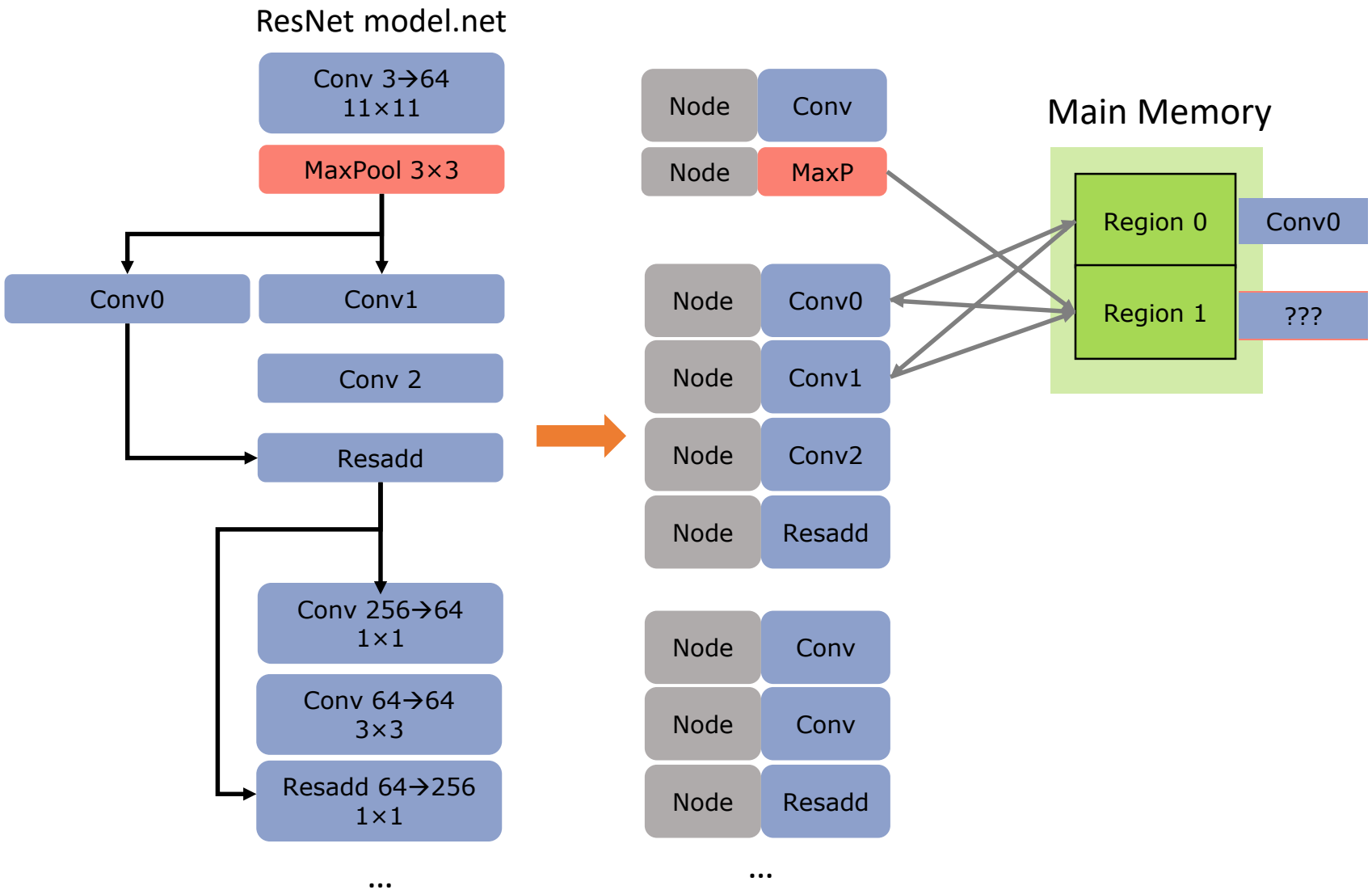
Snowflake compiler steps

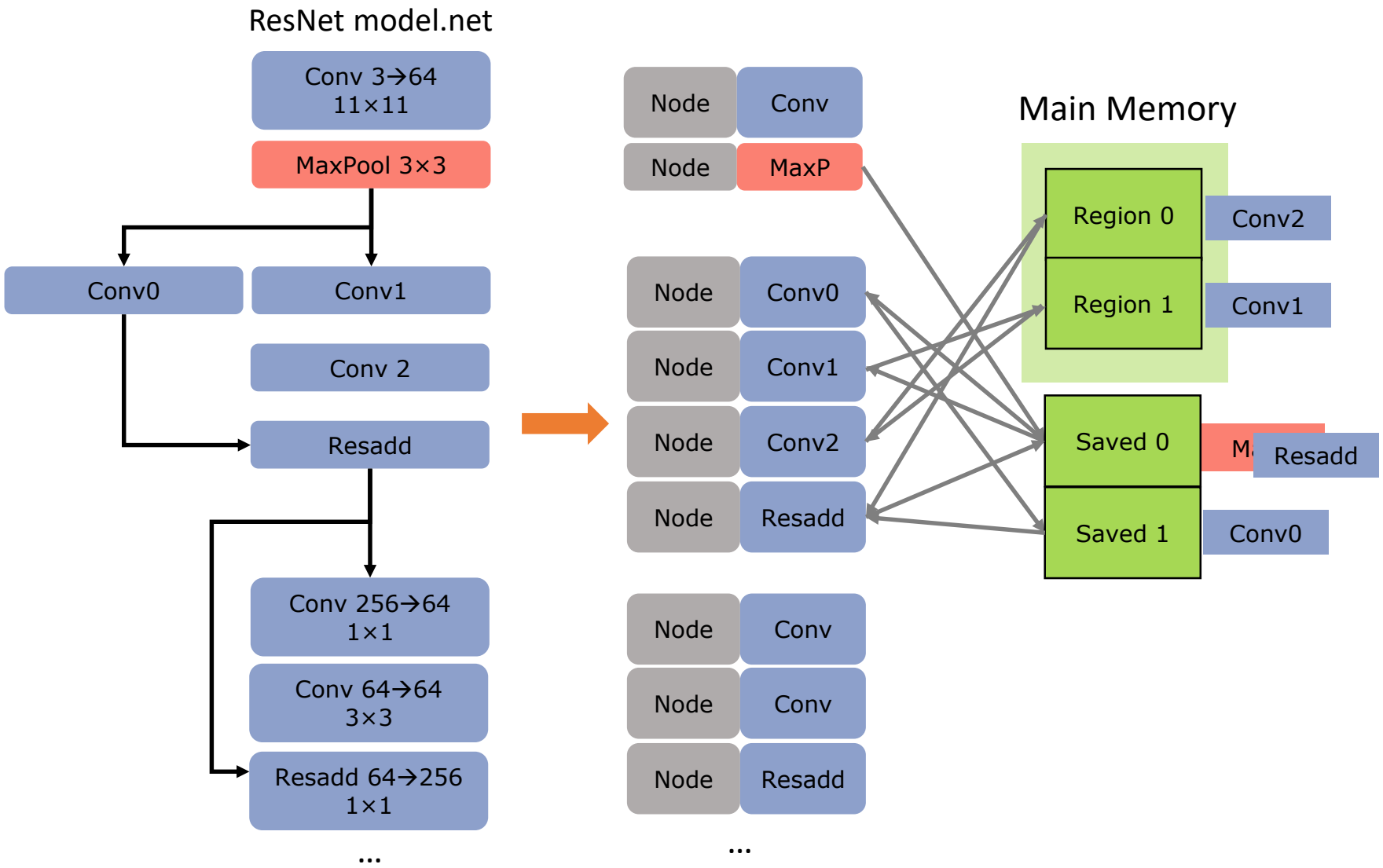
- Parsing
- Partition
- Code generation

Parse

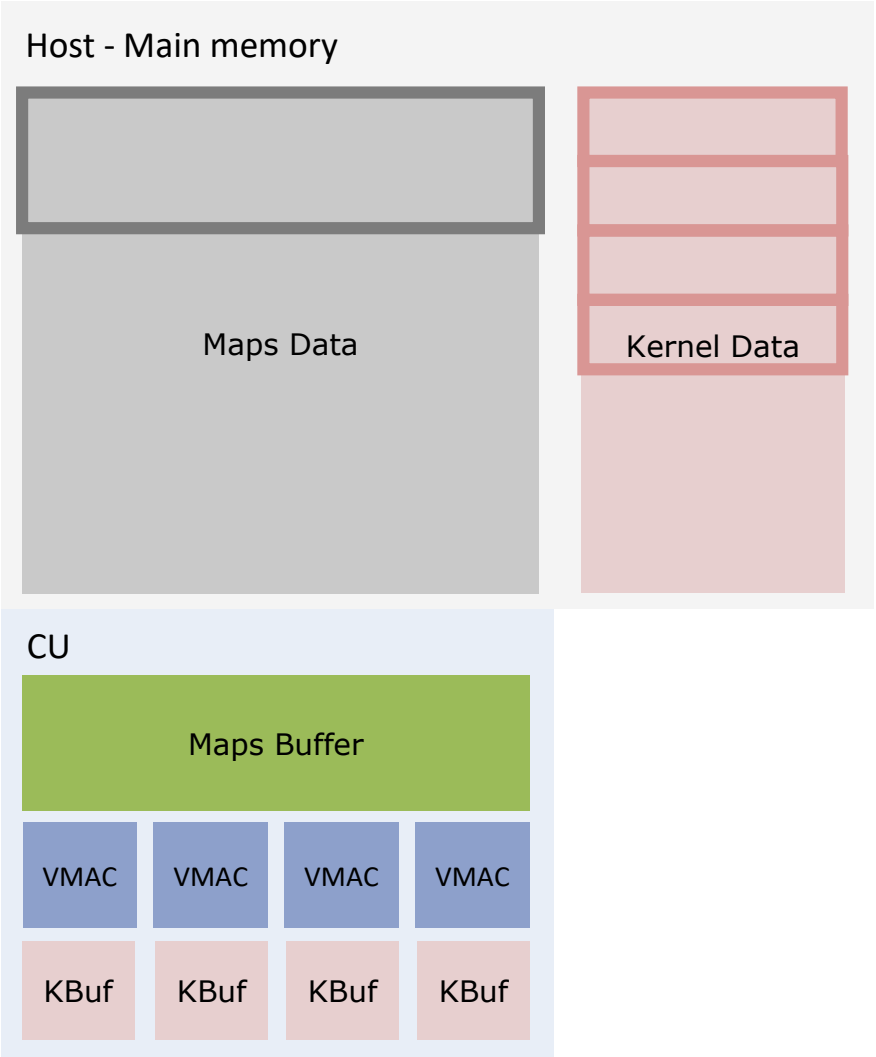


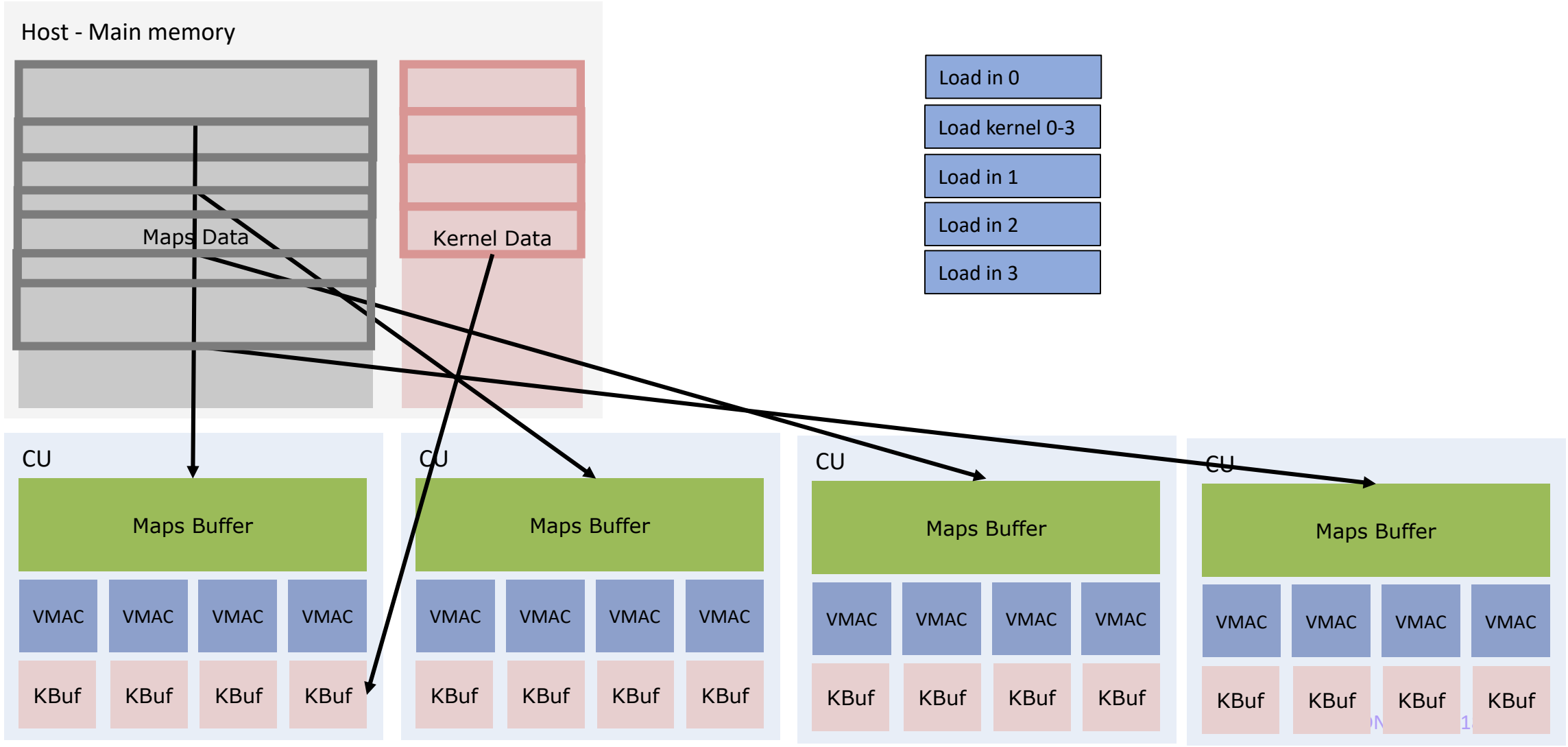
AlexNet model.net


ResNet model.net
Conv 3→64
11×11
MaxPool 3×3
Conv0
Conv1
Conv 2
Resadd
Conv 256→64
1×1
Conv 64→64
3×3
Resadd 64→256
1×1
...
Node
Conv
Node
MaxP
Node
Conv0
Node
Conv1
Node
Conv2
Node
Resadd
Node
Conv
Node
Conv
Node
Resadd
...
Main Memory
Region 0
Region 1
Conv0
???

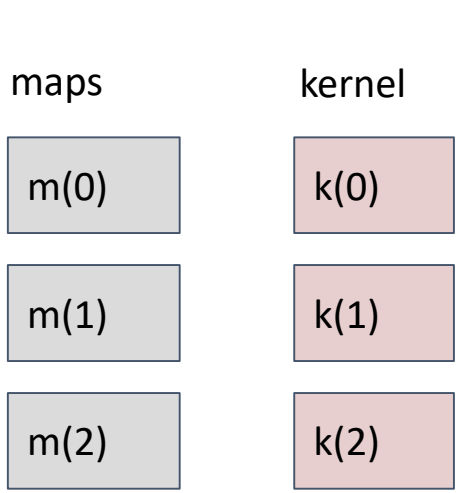

ResNet model.net
Conv 3→64
11×11
MaxPool 3×3
Conv0
Conv1
Conv 2
Resadd
Conv 256→64
1×1
Conv 64→64
3×3
Resadd 64→256
1×1
...
Node
Conv
Node
MaxP
Node
Conv0
Node
Conv1
Node
Conv2
Node
Resadd
Node
Conv
Node
Conv
Node
Resadd
...
Main Memory
Region 0
Region 1
Saved 0
Saved 1
Conv2
Conv1
Resadd
Conv0

Partition




Host - Main memory
Maps Data
Kernel Data
Load in 0
Load kernel 0-3
Load in 1
Load in 2
Load in 3
CU
Maps Buffer
VMAC
VMAC
VMAC
VMAC
KBuf
KBuf
KBuf
KBuf
CU
Maps Buffer
VMAC
VMAC
VMAC
VMAC
KBuf
KBuf
KBuf
KBuf
CU
Maps Buffer
VMAC
VMAC
VMAC
VMAC
KBuf
KBuf
KBuf
KBuf
CU
Maps Buffer
VMAC
VMAC
VMAC
VMAC
KBuf
KBuf
KBuf
KBuf

Order




```
for( k=0;k<3;k++ )
  for ( m=0;m<3;m++ )
```

k(0) m(0) k(0) m(1) k(0) m(2)

k(1) m(0) k(1) m(1) k(1) m(2)

...

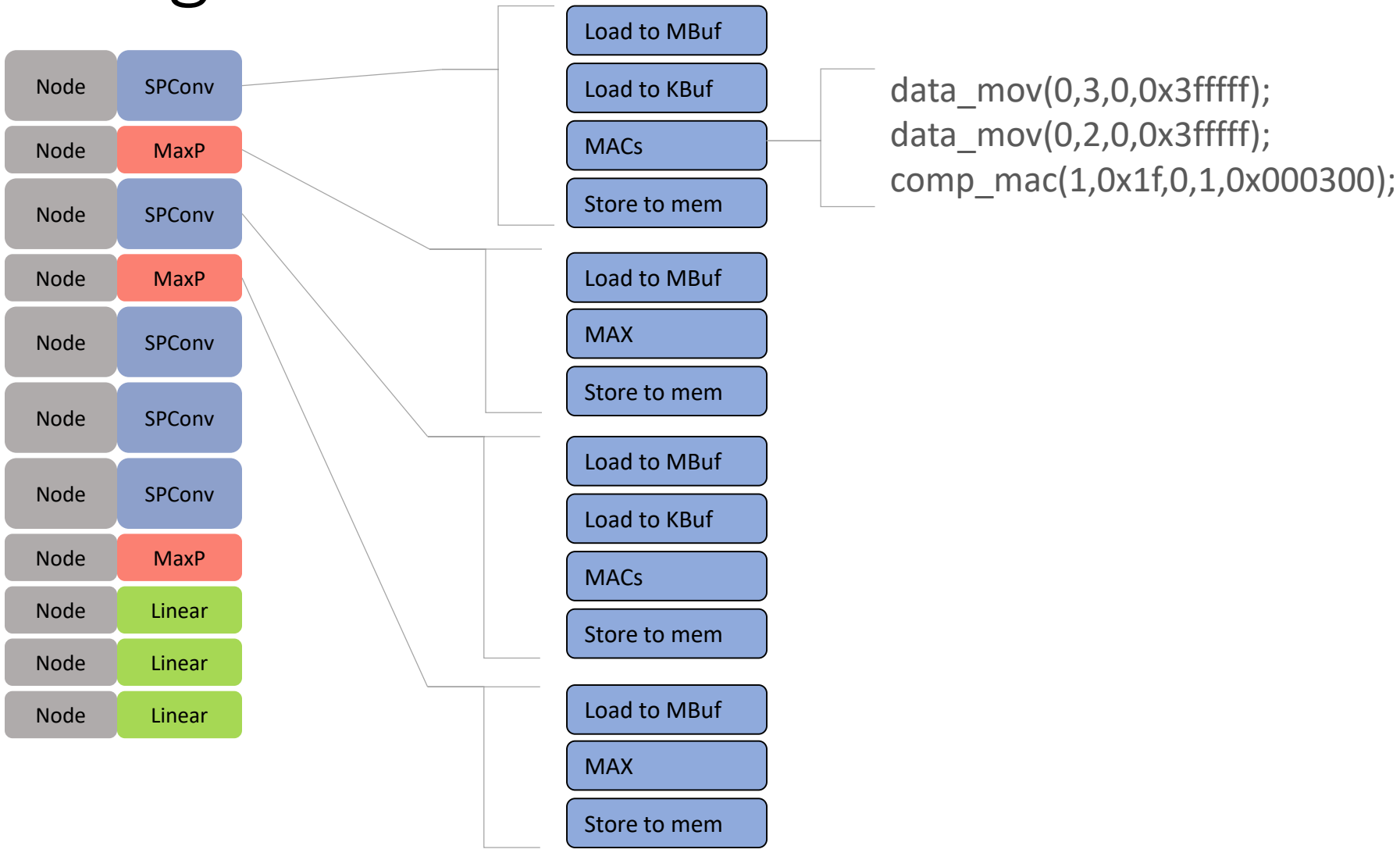
```
for ( m=0;m<3;m++ )
  for( k=0;k<3;k++ )
```

k(0) m(0) k(1) m(0) k(2) m(0)

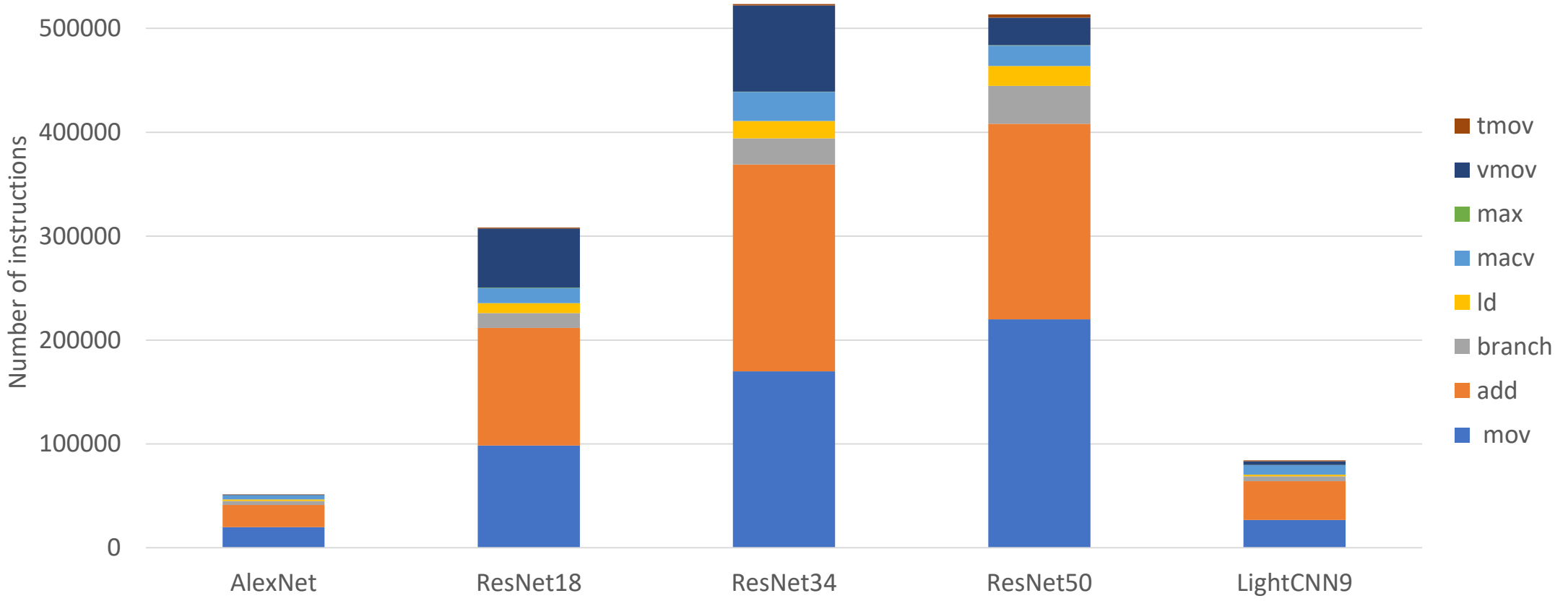
k(0) m(1) k(1) m(1) k(1) m(1)

...

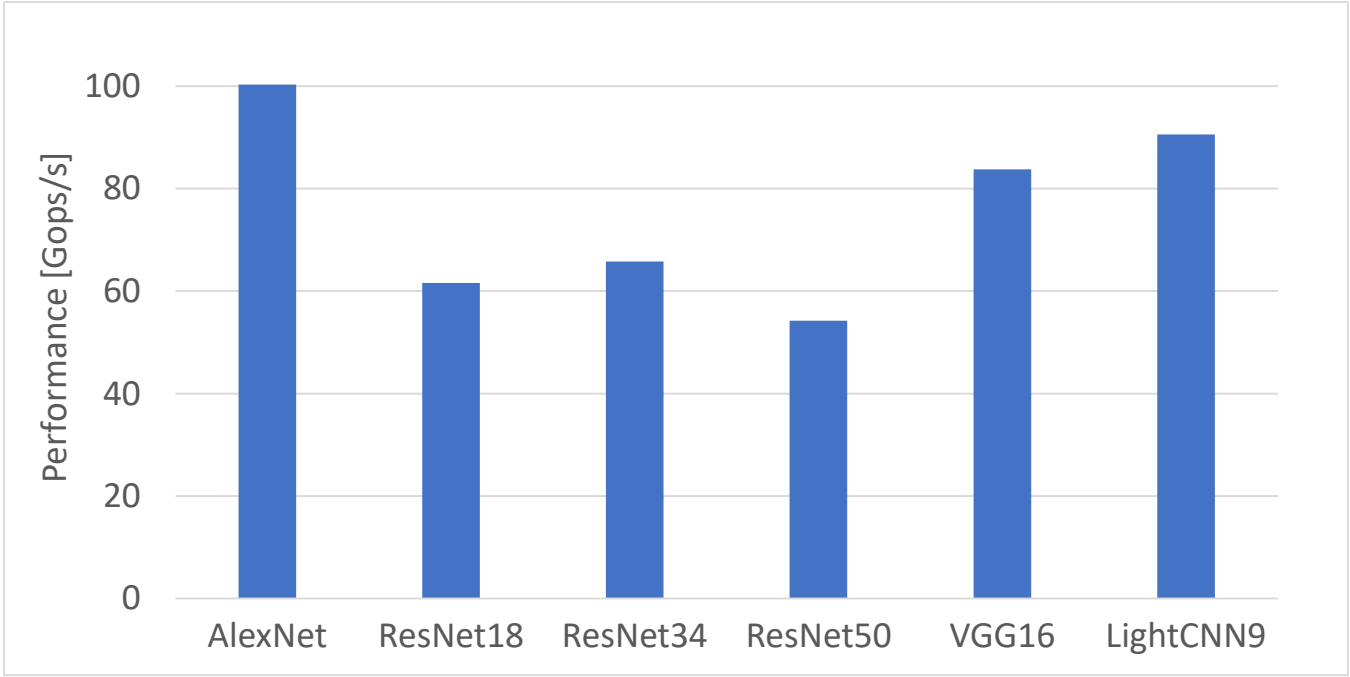
Code generate



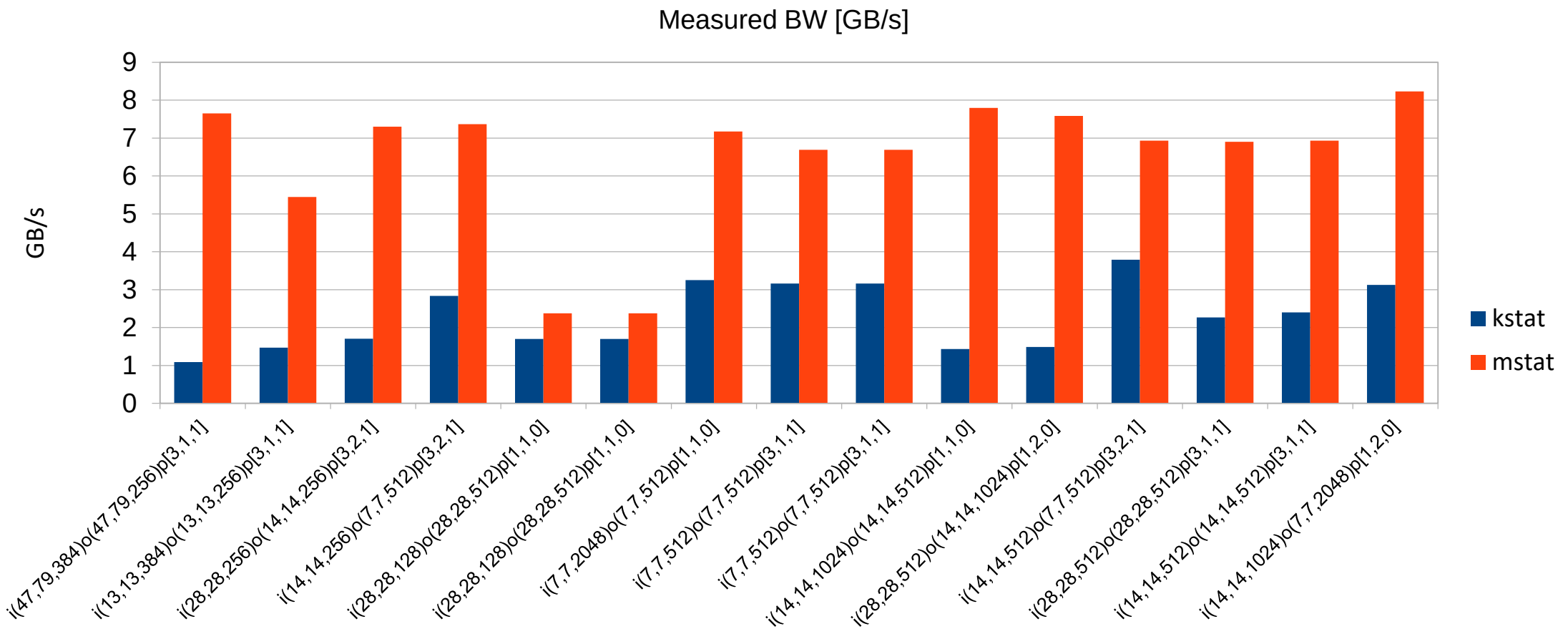
Results – code



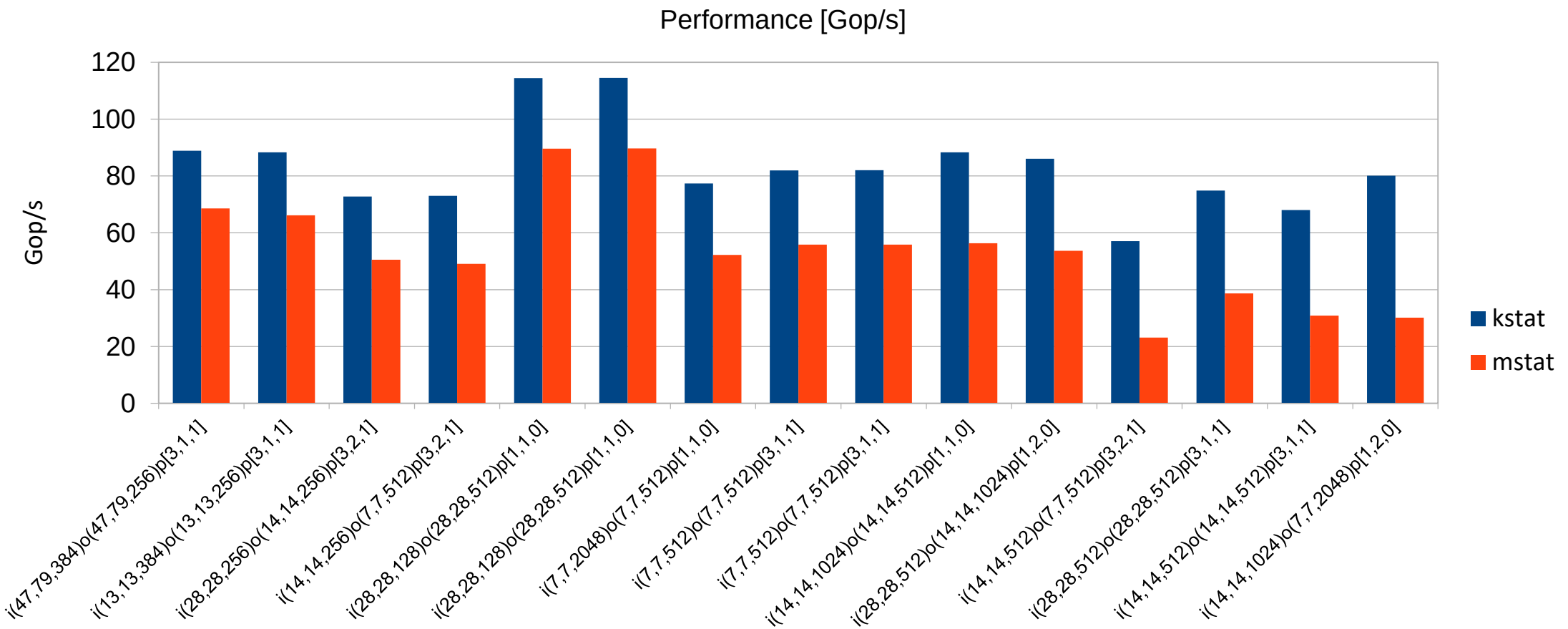
Results - Performance

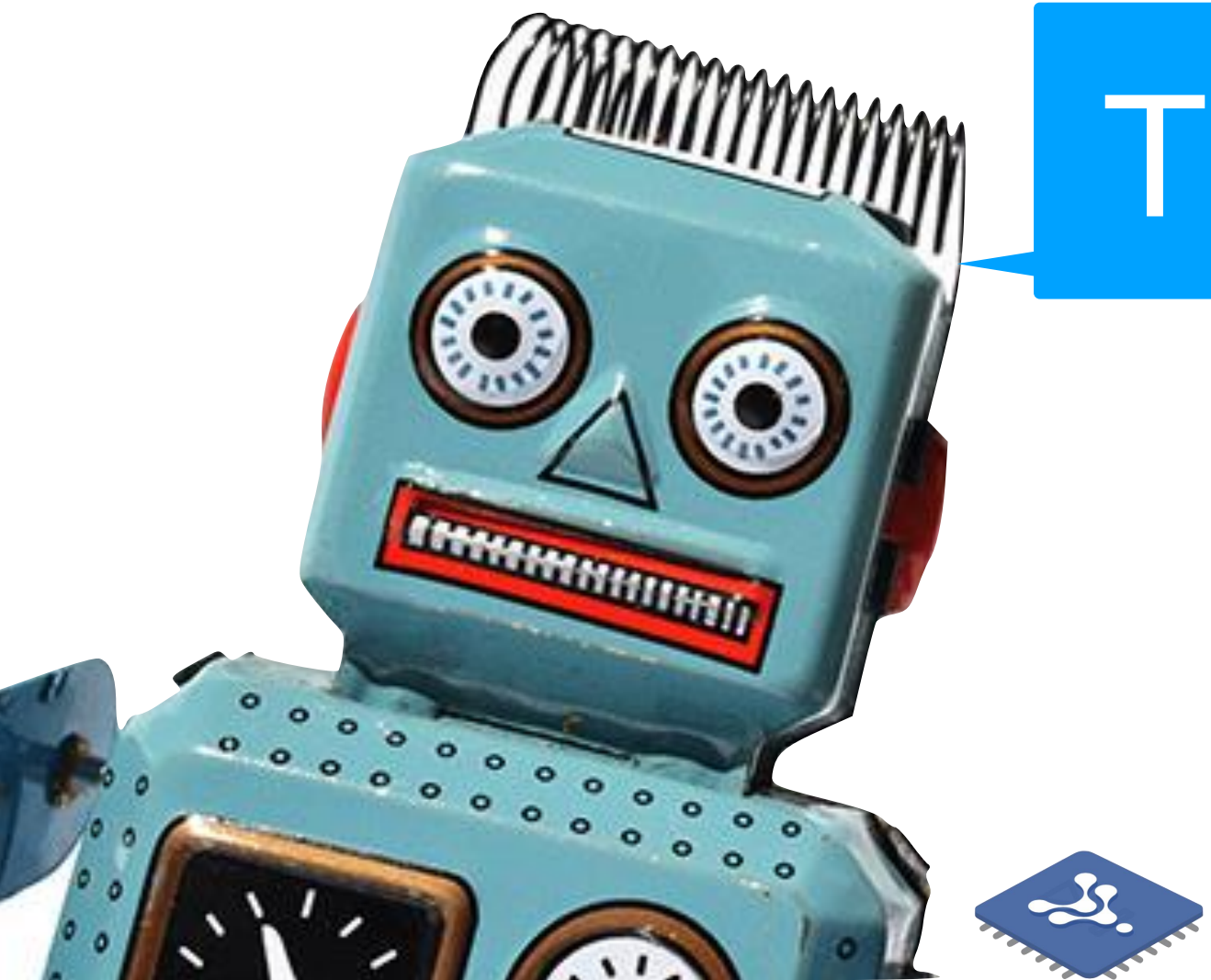


Results Bandwidth



Results





Thank you

FW NXT